MISSING VALUE TREATMENT

Data collection is a initial part for every project. Whenever collecting data some individual will not provide all field information because some field are not mandatory. Let us say Marital_status is an attribute but some people will not enter the details as it is not mandatory.

Treating null values is most important for model building process and data analysis.

How to find Missing Values

In pandas we can check out columns which having missing values.In Pandas missing data is represented by two value:

- None: None is a Python singleton object that is often used for missing data in Python code.
- NaN : NaN (an acronym for Not a Number), is a special floating-point value recognized by all systems that use the standard IEEE floating-point representation

Their are two types of function in pandas to check null values

- isnull()
- notnull()

1. Checking Null Values using isnull function

In [ ]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [ ]:

```
import pandas as pd
```

In [ ]:
```
data = pd.read_csv("/content/drive/My Drive/Selling Project Cont
ent/\
Data Set/Loan_default_classification.csv")

data.head()
```

Out[ ]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Applican
0	LP001002	Male	No	0	Graduate	No	
1	LP001003	Male	Yes	1	Graduate	No	
2	LP001005	Male	Yes	0	Graduate	Yes	
3	LP001006	Male	Yes	0	Not Graduate	No	
4	LP001008	Male	No	0	Graduate	No	

In [ ]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            614 non-null    object 
 1   Gender             601 non-null    object 
 2   Married            611 non-null    object 
 3   Dependents         599 non-null    object 
 4   Education          614 non-null    object 
 5   Self_Employed      582 non-null    object 
 6   ApplicantIncome    614 non-null    int64  
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object 
 12  Loan_Status        614 non-null    object 
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

```
In [ ]: data.isnull()
```

Out[ ]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Applicai
0	False	False	False	False	False	False	
1	False	False	False	False	False	False	
2	False	False	False	False	False	False	
3	False	False	False	False	False	False	
4	False	False	False	False	False	False	
...	...	...	...	...	...	...	
609	False	False	False	False	False	False	
610	False	False	False	False	False	False	
611	False	False	False	False	False	False	
612	False	False	False	False	False	False	
613	False	False	False	False	False	False	

614 rows × 13 columns

Usually we use sum function to add all True values(Which is Null values). So that we can able to find number of null values in each columns

```
In [ ]: data.isnull().sum().sort_values(ascending=False)
```

```
Out[ ]: Credit_History       50
        Self_Employed        32
        LoanAmount           22
        Dependents           15
        Loan_Amount_Term     14
        Gender               13
        Married               3
        Loan_ID               0
        Education             0
        ApplicantIncome       0
        CoapplicantIncome     0
        Property_Area         0
        Loan_Status           0
        dtype: int64
```

Above numbers represents number of null values in each columns. For example Credit_History has 50 Null values, Self_Employed has 32 null values and so on.

2. Checking Null Values using notnull function

In [ ]:
```
data.notnull().sum().sort_values(ascending=False)
```

Out[ ]:
```
Loan_ID              614
Education            614
ApplicantIncome      614
CoapplicantIncome    614
Property_Area        614
Loan_Status          614
Married              611
Gender               601
Loan_Amount_Term     600
Dependents           599
LoanAmount           592
Self_Employed        582
Credit_History       564
dtype: int64
```

Above numbers represents number of values in each columns. For example Loan_Id has 614 values, Education has 614 values and so on.

Normally we use isnull() function to find the number of null values in each columns.

Treating Missing Values

Handling missing values is very important during data preprocessing as many machine learning algorithm will not support missing values.

Different Methods to treat missing values

1. Droping rows or columns having missing values
2. Using Pandas function
3. Impute missing values for categorical Variable
4. Impute missing Values for continuous variable
5. Creating a model to predict missing values

1. Droping rows having missing values

Missing values handled by removing rows or columns which having null values. Columns can be removed when more than 50% of values are null values, and rows are removed when one or more column values having null.

Advantage :

- Model will be trained on dataset which all null values removed creates robust model.

Disadvantage:

- Lots of Information loss
- Accuracy will fall down if we train model on dataset in which no of rows or columns removed is high compared to model trained on dataset in which null values are replaced by values.

In [ ]:
```
data.dropna()
```

Out[ ]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Applic
1	LP001003	Male	Yes	1	Graduate	No	
2	LP001005	Male	Yes	0	Graduate	Yes	
3	LP001006	Male	Yes	0	Not Graduate	No	
4	LP001008	Male	No	0	Graduate	No	
5	LP001011	Male	Yes	2	Graduate	Yes	
...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	
610	LP002979	Male	Yes	3+	Graduate	No	
611	LP002983	Male	Yes	1	Graduate	No	
612	LP002984	Male	Yes	2	Graduate	No	
613	LP002990	Female	No	0	Graduate	Yes	

480 rows × 13 columns

2. Using Pandas function

Their are two ways of filling missing values

1. pad/fill

   - Fill methods Forward. This is known as the Last observation carried forward (LOCF) method.

   ```
   df.fillna(method='pad')
           or
   df['column_name'].fillna(method='pad')
   ```

2. bfill/backfill

   - Fill methods Backward

   ```
   df.fillna(method='backfill')
           or
   df['column_name'].fillna(method='pad')
   ```

3. Impute missing values for categorical Variable

When the missing values is from categorical columns, then missing values is replaced by most frequently occured values. If the missing values is large then it is replaced by Unique values. We will use `mode` function to get frequently used value.

Advantage:

- Prevents data loss.
- Works well on small data set

Disadvantage:

- Addition of new features to the model while encoding, which may result in poor performance.

In [ ]:
```
# Helps to know the data types
data.dtypes
```

Out[ ]:
```
Loan_ID               object
Gender                object
Married               object
Dependents            object
Education             object
Self_Employed         object
ApplicantIncome        int64
CoapplicantIncome    float64
LoanAmount           float64
Loan_Amount_Term     float64
Credit_History       float64
Property_Area         object
Loan_Status           object
dtype: object
```

In [ ]:
```
data['Self_Employed'] = data['Self_Employed'].fillna(data['Self_Employed'].mode()[0])
```

In [ ]:
```
data['Self_Employed'].isnull().sum()
```

Out[ ]:
```
0
```

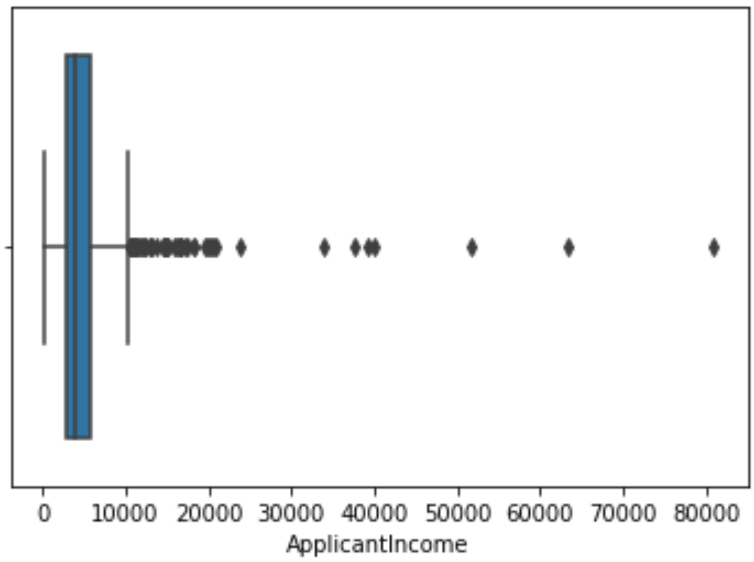
4. Impute missing Values for continuous variable

- For continuous variabable we will replace null values using mean, median and mode. Commonly we will use mean and median for replacing the null values. These two approximation are statistical approach for replacing null values.As if a variable has outliers then we will replace null values with median and if a variable has no outliers then we will replace null values with mean.

In [ ]:
```
import seaborn as sns
sns.boxplot(data['ApplicantIncome'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:4
3: FutureWarning: Pass the following variable as a keyword arg:
x. From version 0.12, the only valid positional argument will be
`data`, and passing other arguments without an explicit keyword
will result in an error or misinterpretation.
  FutureWarning
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8c79cc2950>
```



In [ ]:
```
"""
From above  plot we can concluded that their are more outliers
so that we will use median
"""
data['ApplicantIncome'] = data['ApplicantIncome'].fillna(data['A
pplicantIncome'].median())
```

5. Creating a model to predict missing values

- The regression or classification model can be used for the prediction of missing values depending on the nature (categorical or continuous) of the feature having missing value.